xanaspec-D17
=== 12.01.18

REF

XANADU® TEXT SYSTEMS: PAST AND CURRENT DATA STRUCTURES

Theodor Holm Nelson, Founding Designer, Project Xanadu

TRADEMARKS. Xanadu® and ZigZag® are registered software trademarks of Project Xanadu, Inc. XanaduSpace™, Xanadesk™ and Xanafrac™ are claimed trademarks of Project Xanadu.

These specifics are set forth here for anyone who wants to understand the Xanadu ideas in depth and today's ongoing Xanadu implementations.

We expect that you already know the basic ideas. Just to make sure, look at our little demo on Youtube-- "Ted Nelson demonstrates Xanadu Space." See also our document "Xanadu Connected Literary Structure", which is currently up to draft 9 (Xanalit-D9).

The design below is a simplified version of the hyperoptimized Xanadu designs of the 1980s, which almost nobody could understand. It has been adapted to today's file systems.

This is intended to show the basic data structures, which may seem complex but stem from one simple idea: QUOTED CONTENT WITH OVERLAYS. By breaking a document out into quoted content and separate overlaid links, extraordinary things are possible.

If you watch the demo, and understand our intentions, the actual internal operations are fairly obvious.

In what follows, we will say "xanadoc" and "xanalink" at all times, rather than "document" and "link," to avoid all possible confusion with conventional electronic documents and weblinks.

=== === === HOW A XANADU SYSTEM WORKS

>>> This section covers old and current Xanadu implementations-- including XanaduSpace, XanaDesk and Xanafrac, as well as the classic version described in "Literary Machines" (xu88 or "Udanax Green", and the disastrous Autodesk version ("Udanax Gold").

A xanadoc is assembled from stabilized content portions selected by an Edit Decision

List (EDL). By "stabilized" we mean that the content must be guaranteed not to change.

To assemble the xanadoc, these content portions are brought in and concatenated, then overlaid by links and formatting. The links and formatting are brought in by the Overlay Decision List (ODL).

For presentation, these formatted pages are shown connected by transclusion beams (showing identities of content) and link beams (showing link connections).

Publishing a Xanadoc means making available all its constituent parts, which are assembled in the client (as today's complex web pages are).

That is the entire concept. The rest is detail.

What does this accomplish?

- Visible connections create a new dimensionality of literature.

- Xanalinks do not break as xanadocs are edited.

- Every portion of content remains connected to its original context.

- Anyone can make visible links that connect to contents inside a xanadoc, for instance detailed annotation.

Annotation, collaboration, mashups, tagging, management of large-scale document sets, and much more-- may all be implemented and combined easily, far more easily and transparently than with HTML.

<<< [This ends the part shared by all Xanadu versions.)

=== === === NEW, SIMPLIFIED VERSIONS; SKIPPING OPTIIZATION FOR NOW

We are working on several new implementations, simplified for easier implementation in today's environment and now mapped to conventional files.

What follows are the basic technicalities, set up for these new specific files.

We are not discussing optimization. The important thing is to show the logic. Certain optimizations are obvious, but here we are showing the main data structures and what they do in the full unfolding of a xanadoc.

=== === === MULTI DOCUMENTS OPEN

Digitized by the Internet Archive
in 2025

https://archive.org/details/pastcurrentdatas00tedn

It is intended that portions of numerous documents can be open at the same time. Management of their portions will be an issue.

=== === === MICROVERSIONING

A xanadoc may have a version designator, which exactly specifies the state of the document being referred to. (Explained below.)

This is a way to refer to any given microversion of any document.

=== === === THE OPERATIONS

We will not discuss the operations in detail, but present the data structures necessary to carry them out.

Every operation deals with these structures and only these structures.

These operations include
- opening and closing documents and windows, scrolling up and down
- editing (insert, delete, transclude-- we do not support "cut" and "paste", which lose the origins of content)
- showing xanalink beams
- showing transclusion beams

The specific operations are fairly obvious, based on the data structures to be described below. We are not detailing the operations here, but they are fairly straightforward when you understand the data structures.

=== === === THE SPECIFIC DATA STRUCTURES: MOST DON'T CHANGE

There are a number of Xanadu data structures.

Most are cumulative and do not change except for new appended content.

Only two are changeable.

=== === === UNCHANGEABLE DATA FORMATS

=== SOURCEDOCS: UNCHANGING CONTENT SOURCES (immutable, extension .txt)

Content sent for must be guaranteed not to change, and all strings must keep their same

address. This is not a technical issue so much as a political and policy issue.

It may be necessary to store and publish source content permanently as text sourcedocs. But there are alternatives.

Third-party Sourcedocs which are trusted to be stable (for instance, Project Gutenberg) may be used. Publishers may source their own stable content, or (for instance) put it in the Internet Archive.

= OTHER SOURCEDOCS

Source files are initially assumed to be text.

At a later time we may use lump files of other conventional types with embedded markup, e.g.--
- .pdf
- .html
- .doc

However, when content is used from files with markup, only the visible characters will be counted. All the markup in these sourcedocs is ignored or converted to xanalinks.

= STEPS OF PERMANIZING AN HTML SOURCEDOC

It should work like this--

When a user starts to use part of an existing web page (transcluding from it or linking to it)--
- cache that page in original form, with URL and date
- extract and permanize the user-visible text-- strip the codes, thus assigning successive addresses to the exposed positions of text content

= LATER AUDIO AND VIDEO

Later content sources will include audio and video files, similarly stabilized.

The code for transclusion and linking should be very similar to that for text, though of course the interface must be extended for the purpose.

=== XANALINK (immutable, extension .xanalink)

A xanalink is an individual link or formatting unit, separate from the content. (This is

sometimes called "standoff markup" or "offset markup".)

Xanalinks represent everything which is not in the text itself.

Xanalinks are applied to the content of a xanadoc
- to show visible linkage connections
- for a variety of other functions
- for fonts and appearance

A xanalink has a unique ID, is individually addressable, and is published individually as a first-class object. Every public xanalink is published by itself on the net, and reachable by anyone.

Any published xanalink may be included in any xanadoc.

Xanalinks do not change, but they may be replaced.

(LATER we intend to add link versioning, allowing a link to be changed and versioned while retaining its identity.)

A xanalink has--
- an ID, by which it can be found.
- a TYPE (discussed below)
- a number of ARMS (FACETS, PRONGS or SIDES) with specific meanings. For instance, a comment link has two arms or facets, attached to two areas of text-- the content commented on, and the comment, usually in another document.
- one or more spans per arm or facet (a span is a series of consecutive characters) that the link attaches to. For instance, a comment may have two spans of commented-on text, both of which are being commented on at once.

ORDINARILY a span points to characters, but can point to any unit, e.g. whole document or another link.

SIMPLE EXAMPLES OF LINK TYPES

There will be many more link types; these are a few to begin with.

- title (points to a span of text, preferably within the document)
- heading (points to a span of text)
- paragraph (points to the character beginning the paragraph)
- newpage / page break (points to the character beginning the new page)
- comment (2-sided-- arm 1: Commented-On text; arm 2: comment)
- disagreement (1 or 2-sided-- arm 1: Text-Disagreed-With; arm 2: Disagreement-Text (optional))
- endorsement (1 or 2-sided-- arm 1: Endorsed-Text; arm 2: Endorsement-Text (optional))

- typeless 1-sided (equivalent to a blank sticky note)
- typeless 2-sided (2-sided, connecting two different passages)
- apply another ODL (for instance, for typesetting formats)
- derivation link (pointing to documents of origin)

= XANALINKS WHICH CAN BE SIMPLY OVERLAID

Overlay xanalinks fit right over the typeset (as seen in the XanaduSpace demo on YouTube). They do not break the topology and can be seen in simple overlay mode--

- comment (2-sided)
- disagreement
- endorsement
- typeless 1-sided (equivalent to a blank sticky note)
- typeless 2-sided (connecting two different passages)

= XANALINKS WHICH STRETCH AND CUT LAYOUT

The effect of some xanalinks is to stretch the layout, creating gaps in the text. Examples:
- title
- heading
- paragraph

The effect of other xanalinks is to cut the layout. Example:
- newpage (page break)

= XANALINKS WHICH DISRUPT THE TOPOLOGY

Some xanalinks may disrupt the topology in other ways (e.g. illustration, hole in the page, interactive button, etc.)

We will not deal with those here.

= XANALINK FORMATS
[name and extension already specified]
type= XXX
endset1=
span
span
...
endset2=
span
span
...

- typeless 1-sided (equivalent to a blank sticky note)
- typeless 2-sided (2-sided, connecting two different passages)
- apply another ODL (for instance, for typesetting formats)
- derivation link (pointing to documents of origin)

= XANALINKS WHICH CAN BE SIMPLY OVERLAID

Overlay xanalinks fit right over the typeset (as seen in the XanaduSpace demo on YouTube). They do not break the topology and can be seen in simple overlay mode--

- comment (2-sided)
- disagreement
- endorsement
- typeless 1-sided (equivalent to a blank sticky note)
- typeless 2-sided (connecting two different passages)

= XANALINKS WHICH STRETCH AND CUT LAYOUT

The effect of some xanalinks is to stretch the layout, creating gaps in the text. Examples:
- title
- heading
- paragraph

The effect of other xanalinks is to cut the layout. Example:
- newpage (page break)

= XANALINKS WHICH DISRUPT THE TOPOLOGY

Some xanalinks may disrupt the topology in other ways (e.g. illustration, hole in the page, interactive button, etc.)

We will not deal with those here.

= XANALINK FORMATS
[name and extension already specified]
type= XXX
endset1=
span
span
...
endset2=
span
span
...

endset3=
...

=== === === CUMULATIVE DATA STRUCTURES (read-and-append)

=== PERMASCROLLS (append-only, extension .pscr)

A permascroll is a text sourcefile set up to receive new content from an author or group.

The addresses of content already written on the permascroll do not change though it is continually added to.

As a sourcefile, it supplies the already-existing content as requested, with unchanging addresses. Meanwhile, it appends newly-arriving inserted content, whose addresses do not change once posted.

=== THE OPLIST (append-only, extension .xanops)

Our version management (optional) allows branching versions in a structure that is treelike or even cross-connected. (For an earlier implementation, see http://xanadu.com.au/ted/OSMIC/ .)

The opList tells how to get to any particular version, rebuilding its EDL and ODL. The OpList is cumulative (append-only).

The system can re-enact prior versions and branching versions, showing any two versions connected.

Each edit operation is saved with a time stamp and the parameters required to act it out.

An operation may affect the EDL, the ODL, or both.

The edit operations are--
- begin new document
- insert new content
- delete
- transclude (make virtual copy of content elsewhere)
- move text (transclude with deletion)
- make link
- replace link
- delete link

"Copy" and "paste" are not supported, since they lose the origins of content.

Each operation creates a new microversion. These may be reenacted to take different branches and assemble different versions for comparison or re-use.

A microversion designator tells what operations to re-enact, starting with the first operation. For example,

13a42a7b16

tells what steps to enact--
- the first 13 operations
- then take branch a after the 13th operation
- then enact 42 more operations
- take another branch a from that state
- then enact 7 operations
- then take a branch from that state-- not the first branch, a, but a second branch, b
- enact 16 more operations

= OPLIST FORMAT

The format for the stored operation is an Opname followed by its parameters, as follows--
- beginDoc (begin new document, no parameters)
- insertNew (insert new text-- destination slit, new string, returns permascroll address to EDL)
- deleteText (concaspan-- starting sit, ending slit)
- transcludeText (make virtual copy of content elsewhere (concaspan-- starting slit, ending slit, destination slit)
- moveText (move text, transclude with deletion; concaspan-- starting slit, ending slit, destination slIt)
- makeLink (make link-- type, facet concaspans)
- replaceLink (replace link-- which to replace, type, facet concaspans)
- deleteLink (delete link-- which to delete)

=== === === XANALINKS, UNCHANGING UNITS (extension .xanalink)

Xanalinks are not changed, only deleted or replaced.

XANALINK OPERATIONS (as further specified above)
- make link
- replace link
- delete link

(Later specs will allow changing xanalinks to be versioned.)

Each operation creates a new microversion. These may be reenacted to take different branches and assemble different versions for comparison or re-use.

A microversion designator tells what operations to re-enact, starting with the first operation. For example,

13ad:2a7bt6

tells what steps to enact--
- the first 13 operations
- then take branch a after the 13th operation
- then enact 42 more operations
- take another branch a from that state
- then enact 7 operations
- then take a branch from that state-- not the first branch, a, but a second branch, b
- enact 16 more operations

= OPLIST FORMAT

The format for the stored operation is an Opname followed by its parameters, as follows--
- beginDoc (begin new document, no parameters)
- insertNew (insert new text-- destination slit, new string, returns permascroll address to EDL)
- deleteText (concaspan-- starting slit, ending slit)
- transcludeText (make virtual copy of content elsewhere (concaspan-- starting slit, ending slit, destination slit)
- moveText (move text, transclude with deletion; concaspan-- starting slit, ending slit, destination slit)
- makeLink (make link-- type, facet concaspans)
- replaceLink (replace link-- which to replace, type, facet concaspans)
- deleteLink (delete link-- which to delete)

=== === === XANALINKS, UNCHANGING UNITS (extension .xanalink)

Xanalinks are not changed, only deleted or replaced.

XANALINK OPERATIONS (as further specified above)
- make link
- replace link
- delete link

(Later specs will allow changing xanalinks to be versioned.)

=== === === CHANGEABLE DATA FORMATS

=== XANADOC FILE (extension .xanadoc)

The xanadoc file is a designator for a whole document or project. (It is expected, however, that many extra xanadoc files will be created, for projects discarded or vacated.)

The xanadoc file specifies the contents and the links of a document, naming the lists by which the document is accessed and assembled.

Its filename does not change but its contents may.

The name of the xanadoc should not be the title of the project, since titles and purposes of a projec frequently change. We recommend an authorname or groupname and a project number.

Example.

IrvingSnerd's47.xanadoc

CONTENTS OF THE XANADOC FILE

The xanadoc file is a list enumerating the parts that make up the document--
- the specific EDL
- the specific ODL
- the specific OpList, if any

These are separately stored from the xanadoc file, to allow multiple access and re-use.

=== === === EDL AND ODL FILES (extensions .xanedl and .xanodl)

The lists for building the document are the EDL and ODL.

These lists are likely to change as the document evolves.

Generally we expect an EDL or ODL to have the same name as the xanadoc file, but sometimes they may not.

Example of the same name used for the four parts--

IrvingSnerd's47.xanadoc
IrvingSnerd's47.xanedl
IrvingSnerd's47.xanodl

IrvingSnerd's47.xanops

=== EDL (Edit Decision List, file type .xanedl (tentative))

The EDL is a list of textual content for building the xanadoc, telling what content to bring in and concatenate. Each element of the list is a spanpointer, pointing at a specific consecutive span of text content. That span may be local or on the net.

Changes to a document do not change the pre-existing content, but they change the list of that content (the EDL) and may add newly-written content to the sourcedocs.

An EDL is a changeable list of unchanging content. The content it points to must be guaranteed not to change. (See Sourcedocs, above.)

Ordinarily an EDL will have the name of its parent document (example above), but this is not required. For example, another document may apply a different set of overlays to the same EDL.

=== ODL (Overlay Decision List, file type .xanodl (tentative))

Ordinarily an ODL will have the name of its parent document.

The ODL tells what overlays to apply to the content. It is list of xanalinks and their addresses. It does not contain the xanalinks themselves.

The editing process can add or delete xanalinks from the ODL, but it does not delete them from published storage, or change them once they are written.

(LATER: we hope to version them, making a xanalink also a cumulative object.)

The ODL can be in any order, since xanalink addresses can go anywhere. A xanalink at the end of the list can point to content at the beginning of the document; indeed, a xanalink anywhere in the ODL can point to any part of any document anywhere.

Ordinarily a xanalink is expected to connect at least one arm to the xanadoc in which it resides, but this is not required; a xanalink may in principle connect other documents without itself being connected to them..

=== === === REFERRING TO XANADU STRUCTURES

These data mechanisms give us an exact way to refer to a xanadoc, every part of a xanadoc, and every version of a xanadoc.

= REFERRING TO A XANADOC

The xanadoc has a universal ID (xanadoc name). For instance, an unqualified reference with no microversion is to the author's designated published version.

IrvingSnerd's1714.xanadoc

REFERRING TO A XANADOC WITH A MICROVERSION

How we refer to a given version of any document: we insert a microversion designator, e.g.

IrvingSnerd's1714_37a134a3.xanadoc

In this example, the microversion is specified by a character string (37a134a3) representing a specific position in the branching graph of microversions.

= REFERRING TO A CONTENT LIST (EDL)

The xanadoc file specifies a default EDL, generally the most recent. Thus we get the EDL specified by the xanadoc file.

E.g.

IrvingSnerd's1813.xanedl

REFERRING TO AN EDL WITH A MICROVERSION

A microversion may be specified for an EDL. For instance, the xanadoc file may specify a particular microversion as the one to be downloaded:

IrvingSnerd's1714_37a134a3.xanedl

This tells the Xanadu client fo build microversion 37a134a3.

= REFERRING TO AN OVERLAY LIST (ODL)

The xanadoc file specifies a default ODL. Thus we get the ODL specified by the xanadoc file.

E.g.

IrvingSnerd's1813.xanodl

REFERRING TO AN OVERLAY LIST WITH A MICROVERSION

A microversion may be specified for an ODL in the same way:

IrvingSnerd's1714_37a134a3.xanodl

= REFERRING TO A XANALINK

E.g.

IrvingSnerd's1813_795.xanalink

The link's serial number shows that it is xanalink 795 of the xanadoc "IrvingSnerd's1813". Xanalinks are numbered in simple ascending sequence.

Xanalinks are not currently versioned.

= REFERRING TO A SPAN OF CONTENT FROM A SOURCEDOC (spanPtr)

The Xanadu spanpointer is a constituent of an EDL. Since a sourcedoc is fixed and consecutive, it specifies a start and length within a URL.

url, offset, length

= REFERRING TO A SPAN OF CONTENT FROM CONCATENATED TEXT

Users will often select spans of content from a page. This concatenated text (concatext) of a document may come from a variety of sourcedocs.

This means a user's consecutive selection from concatext will typically NOT be specified by one spanpointer. It is likely to resolve to a number of spanpointers with new outer boundaries--

url, new offset, newlength
url, offset, length
url, offset, length
...
url, new offset, newlength

= REFERRING TO A PAGE IN A XANADOC

It will be possible to refer to a page in a xanadoc by a link to the newpage link itself.

Another strategy is to refer to content on the page (since that is often what you want to refer to. In that case you specify concatext--

url, offset, length
url, offset, length
url, offset, length
...

=30=

Another strategy is to refer to content on the page (since that is often what you want to refer to. In that case you specify context--

url, offset, length
url, offset, length
url, offset, length
...